5页面置换、请求分段

2018年11月21日　14:20

-
- 页面置换算法

1. 页面置换算法(Page-Replacement Algorithms)：选择换出页面的算法

一. Optimal和FIFO

1. **最佳(Optimal)**置换算法：选择以后永不使用/最长(未来)时间内不再访问的页面
   1) 由Belady于1966年提出的一种理论上的算法，通常可保证获得最低的缺页率
   2) 因为是“向后看”的，无法实现，只用来评价其它算法
   3)

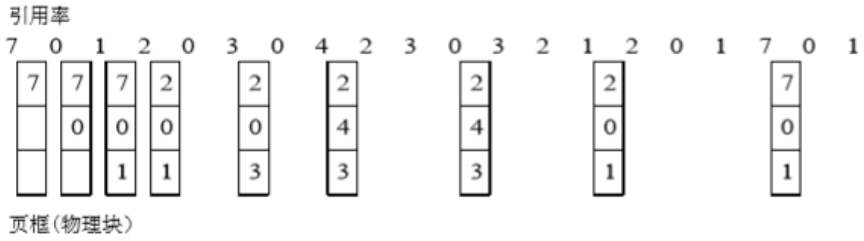


图 4-26　利用最佳页面置换算法时的置换图

2. **先进先出(FIFO)**页面置换算法：选择最先进入内存的页面/驻留时间最久的页面
   1) 是最早出现的置换算法
   2) 需要把调入内存的页面依次排成队列，设置一个替换指针指向最老的页面
   3) 可能淘汰有全局变量、常用函数、例程的页面
   4)

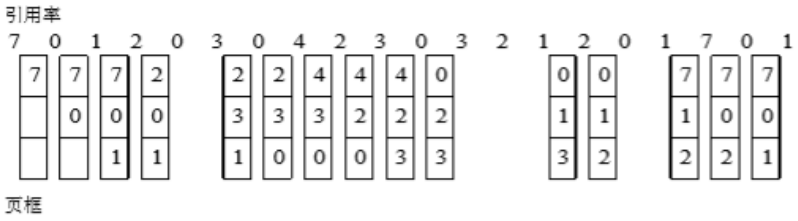


图 4-27　利用 FIFO 置换算法时的置换图

二. **最近最久未使用(Least Recently Used)**置换算法和LFU

1. LRU的描述
   1) 选择(距离上一次使用的)时间最长的页面
   2) 需要每个页面中添加一个字段记录上次使用以来的时间t
   3) 是“向前看”的较好算法，不过过去和未来的走向并无必然联系
   4)

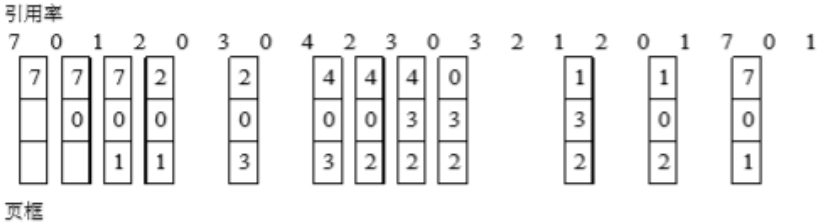


图 4-28　LRU 页面置换算法

2. LRU的硬件支持
   1) 寄存器
      (1) $R = R_{n-1}R_{n-2}R_{n-3} \cdots R_2R_1R_0$
      (2) 访问某物理块时将其R_{n-1}置1

(3) 每隔一段时间（如100ms）将所有寄存器右移一位

(4) 最小二进制数值的寄存器对应的页面就是淘汰页面，如第3行

(5)

实页 \ R	R_7	R_6	R_5	R_4	R_3	R_2	R_1	R_0
1	0	1	0	1	0	0	1	0
2	1	0	1	0	1	1	0	0
3	0	0	0	0	0	1	0	0
4	0	1	1	0	1	0	1	1
5	1	1	0	1	0	1	1	0
6	0	0	1	0	1	0	1	1
7	0	0	0	0	0	1	1	1
8	0	1	1	0	1	1	0	1

图 4-29 某进程具有 8 个页面时的 LRU 访问情况

2) 栈

(1) 每当访问某页面时，将其页面号移出，压入栈顶

(2) 栈底的就是淘汰页面

(3)

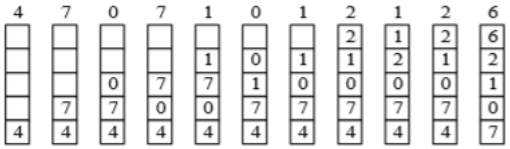


图 4-30 用栈保存当前使用页面时栈的变化情况

3. **最少使用**(Least Frequently Used)置换算法：选择最近时期内使用最少的页

1) 1ms可能访问页面成千上万次，因而难以实现记录访问次数

2) 只能用LRU寄存器反映某段时间为单位的使用频率

3) 并不能真实反映使用频率，因为时间间隔内访问1次和1000次是等效的

三. **Clock置换/最近未用**(Not Recently Used)

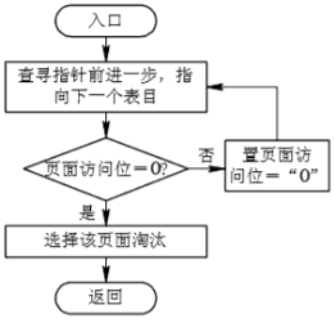
1. 简单的Clock置换

1) 每页设置一访问位，被访问后置1

2) 淘汰时依次找访问位0，并把1置0

3) 内存中的页面被保存成循环链表，因而全为1时会回到第一个做淘汰

4)



块号	页号	访问位	指针
0			
1			
2	4	0	替换指针
3			
4	2	1	
5			
6	5	0	
7	1	1	

图 4-31 简单 Clock 置换算法的流程和示例

2. 改进型Clock置换

1) 除了访问位A以外，还要考虑修改位M

(1) A=0，M=0：该页最近既未被访问，又未被修改，是最佳淘汰页

(2) A=0，M=1：该页最近未被访问，但已被修改，并不是很好的淘汰页

(3) A=1，M=0：该页最近已被访问，但未被修改，该页有可能再被访问

(4) A=1，M=1：该页最近已被访问且被修改，该页可能再被访问

2) 执行过程
   (1) 找A=0且M=0的页面淘汰，（此次查找不改访问位A）
   (2) 找A=0而M=1的页面淘汰，并将此次查找过程经过的A置0
   (3) 回到第一步
3) 尽量减少磁盘I/O次数，但扫描次数可能多了
4) 是UNIX的算法

四. **页面缓冲**算法(Page Buffering Algorithm)

1. 影响页面换进换出效率的若干因素
   1) 页面置换算法，影响了缺页率
   2) 写回磁盘的频率。如把已修改换出页面挂在一个链表上，暂时不写进磁盘，等表长到一定数值时一起写回，就能减少磁盘I/O次数和换出开销
   3) 读入内存的频率。如果未写回磁盘时又想访问已修改页面，可直接从修改换出页面链表上取下，就能减少读入内存的频率，减少换进开销
2. 页面缓冲算法PBA
   1) 为降低页面换进换出频率，VAX/VMS系统中设置了：
      (1) 空闲页面链表，链接了空闲物理块
         i. 读入的页面可以装入该表第一个物理块
         ii. 未修改的换出页面都挂在其后，下次想读，可以直接取下
      (2) 修改页面链表，链接了已修改待换出页面
         i. 想换出已修改的页面时，将该物理块挂在其后
   2) 特点：
      (1) 显著降低页面换进换出频率，减少磁盘I/O次数，减少换进换出开销
      (2) 因换进换出开销大幅减少，可用FIFO等不需硬件支持的置换策略

五. 访问内存的有效时间EAT

1. 需要讨论页表项是否在快表中，不在快表还要讨论页表项在内存中与否，分别用0-1分布数学期望

-
- “抖动”与工作集

1. 抖动(Thrashing)：刚被换出的页面很快又被访问。运行时间大部分花在对换

一. 多道程序度与抖动

1. 多道程序度与处理机的利用率
   1) 利用率一开始随多道程序数增加而急增，进程数到达Nmax后利用率达到最高，之后缓慢下降，进程数多都某一点后利用率趋于0，就是因为抖动
2. 产生抖动的原因：系统中同时运行的进程太多，分配给每个进程的物理块数太少，不能满足进程正常运行的基本要求，使进程频繁缺页，使系统中排队等待页面调进调出的进程数目增加
   1) 因大部分时间用于换进换出，磁盘的有效访问时间也急剧增加

二. 工作集

1. 工作集的基本概念
   1) 1986Denning提出：程序运行的某段时间内，仅访问一部分页面
   2) 称这部分页面为活跃页面
2. 工作集：某段时间间隔Δ内，进程实际需要访问的页面的集合
   1) 把时间t内的工作集记为w(t,Δ)，Δ称为工作集的窗口尺寸windows size
   2) 工作集又可定义为进程在时间间隔(t-Δ,t)中引用的页面的集合
   3) w(t,Δ)是二元函数，是Δ 的非降函数nondecreasing function
      (1) 即$w(t,\Delta) \subseteq w(t,\Delta+1)$

三. 抖动的预防方法

1. 局部置换

1) 不允许从其他进程获得新物理块，因而抖动被限制在小范围，不影响其他进程
2) 然而抖动进程会长期处于磁盘I/O等待队列，延长其他进程缺页中断处理时间，间接延长其他进程访问磁盘时间

2. 把工作集算法融入到处理机调度中
   1) 调度程序从外存调入作业前，先检查各进程在内存中驻留页面是否够多
   2) 为缺页率高的作业增加新物理块，此时不再调入新作业
3. 利用L=S准则调节缺页率
   1) L：发生两次缺页之间的平均时间
   2) S：平均缺页服务时间，即置换一个页面所需的时间
   3) L远>S说明很少缺页，磁盘能力尚未被充分利用
   4) L<S说明频繁发生缺页，且缺页速度已超过磁盘处理能力
   5) L接近S时，磁盘和处理机才能达到最大利用率
4. 选择暂停进程：即减少多道程序的数目
   1) 选择策略：优先级低、并不重要、占块较多、剩余执行时间最多的块

- ◆
- ◆ 请求分段存储管理方式

一. 请求分段中的硬件支持

1. 请求段表机制

1)

段名	段长	段的基址	存取方式	访问字段A	修改位M	存在位P	增补位	外存始址

2) 存取方式。若为两位，则可有只执行、只读、可读写
3) 访问字段A：访问频繁度。供置换算法参考
4) 修改位M：在内存中是否被修改过、供置换页面时参考
5) 存在位P：是否已调入内存。供程序访问时参考
6) 增补位：是否做过动态增长。是请求分段式管理中特有的字段
7) 外存始址：本段在外存中的始址，即起始盘块号

2. 缺段中断机构

1) 类似缺页中断，不过段不定长，稍复杂

2)

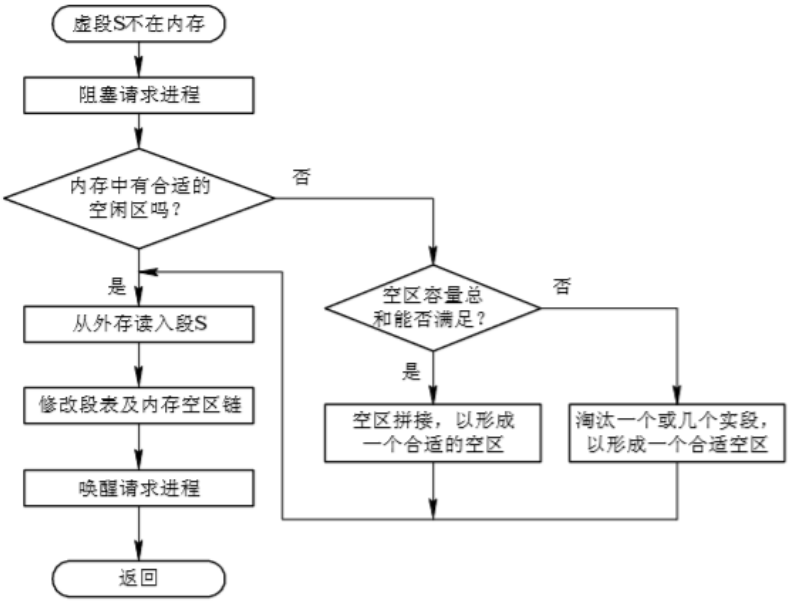


图 4-32 请求分段系统中的中断处理过程

3. 地址变换机构

1) 若要访问段不在内存，必须先调入内存，并修改段表，才能利用段表做地址变换。在传统的地址变换中增加了缺段处理，分段保护处理等

(1)

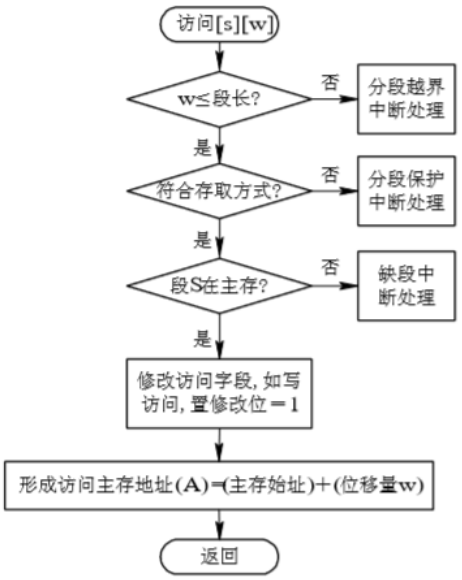


图 4-33　请求分段系统的地址变换过程

二. 分段的共享与保护

1. 共享段表

1)

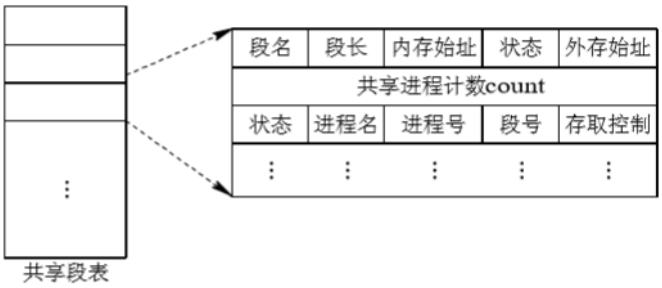


图 4-34　共享段表项

2) 共享计数count：有多少进程在共享该分段

3) 存取控制字段：只执行/只读/可读写

4) 段号：不同进程访问该共享段用到的不同段号

2. 共享段的分配与回收

1) 共享段的分配：

(1) 第一次请求段时在共享段表中增加它的表项，count置1

(2) 之后其他请求段在表项下面填写新一行，count++

2) 共享段的回收

(1) 删除对应表项，count--

(2) 若count减为0，需由系统回收其物理内存，取消其表项

3. 分段保护

1) 越界检查

(1) 利用地址变换机构完成

(2) 若段号大于段表长度，将发出地址越界中断信号

(3) 若段内地址大于段长，也发出地址越界中断信号

2) 存取控制检查

i. 只执行：只能调用该段，不能读内容，不能修改

ii. 只读：只能访问该段的程序/数据

iii. 读/写：运行读，运行写

(1) 既要考虑信息安全，又要满足运行需要

(2) 是基于硬件实现的

3) 环保护机构

(1) 低编号的环有高优先权。如OS核心处于0号环，重要的实用程序和操作系统服务占居中间环，一般应用程序在外环

(2) 程序可以访问驻留在同环或低特权外环的数据

(3) 程序可以调用驻留在同环或搞特权内环的服务

(4)

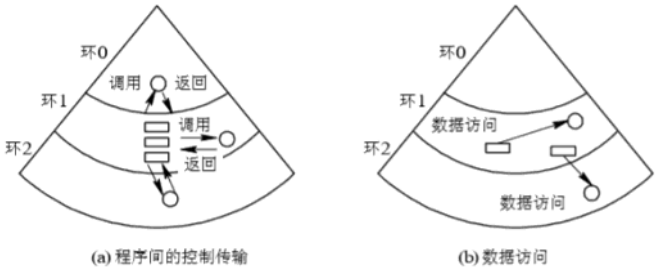


图 4-35 环保护机构

i.

ii.

iii.

iv.

v.

vi.

vii.

viii.

ix.

x.

xi. ---------我是底线-------